A deterministic sparse FFT algorithm for vectors with small support

Gerlind Plonka[*] Katrin Wannenwetsch[†]

April 10, 2015

Abstract

In this paper we consider the special case where a signal $\mathbf{x} \in \mathbb{C}^N$ is known to vanish outside a support interval of length $m < N$. If the support length m of $\mathbf{x}$ or a good bound of it is a-priori known we derive a sublinear deterministic algorithm to compute $\mathbf{x}$ from its discrete Fourier transform $\widehat{\mathbf{x}} \in \mathbb{C}^N$. In case of exact Fourier measurements we require only $\mathcal{O}(m \log m)$ arithmetical operations. For noisy measurements, we propose a stable $\mathcal{O}(m \log N)$ algorithm.

Key words. discrete Fourier transform, sparse Fourier reconstruction, sublinear sparse FFT
AMS Subject classifications. 65T50, 42A38

1 Introduction

It is well-known that FFT algorithms for the computation of the discrete Fourier transform of length N require $\mathcal{O}(N \log N)$ arithmetical operations. However, if the resulting vector is a-priori known to be sparse, i.e., contains only a small number of non-zero components, the question arises, whether we can do this computation in an even faster way. In this paper we derive a new deterministic sparse FFT algorithm for signals $\mathbf{x} \in \mathbb{C}^N$ that are a-priori known to vanish outside a support interval of length $m < N$.

Related work. In recent years many sublinear algorithms for the sparse Fourier transform have been proposed that focus on (approximately) m-sparse vectors or vectors being norm bounded.

Most of the these methods are randomized poly-logarithmic sparse Fourier algorithms achieving e.g. a complexity of $\mathcal{O}(m \log N)$ [5] for m-sparse signals, or even $\mathcal{O}(m \log m)$, see e.g. [11, 12]. However, these algorithms succeed to compute the correct result only with constant probability being smaller than 1. Moreover, there is no efficient method to check the correctness of the result. Another drawback is that

[*]University of Göttingen, Institute for Numerical and Applied Mathematics, Lotzestr. 16-18, 37083 Göttingen, Germany. Email: plonka@math.uni-goettingen.de

[†]University of Göttingen, Institute for Numerical and Applied Mathematics, Lotzestr. 16-18, 37083 Göttingen, Germany. Email: k.wannenwetsch@math.uni-goettingen.de

samples need to be drawn randomly, this is significantly more complicated to realize by hardware. Regarding the numerical results, runtimes start to be more efficient than standard FFT for $m = 50$ and $N > 2^{17}$, see [3, 14] for most of the considered algorithms, and for $m = 50$ and $N > 2^{14}$ for the algorithm in [6]. Another experiment shows for fixed $N = 2^{22}$ that several sparse FFT algorithms start to pay off for $m < 2200$ while the algorithm by [6] is more efficient for $m \leq 2^{17}$, [14]. Newest algorithms are even faster, but the probability to provide the exact result decreases with larger m. For a nice overview of the techniques of randomized sparse Fourier transforms we refer to [3].

Deterministic sparse FFT algorithms proposed e.g. in [1, 2, 9, 10, 11] are applicable to (approximately) sparse or so-called norm bounded signals. Iwen [9, 10] considered sparse m-term approximations of the discrete Fourier transform and achieved an ℓ^2, $\ell^1/\sqrt{m}$ error bound with $\mathcal{O}(m^2 \log^4 N)$ operations. Similar runtime is achieved for the deterministic algorithm in [11], where accessibility not only to the usual signal samples but also to time-shifted samples of the input function is assumed. The deterministic algorithm in [2] for signals being norm bounded by $\sqrt{m}$, the evaluation of δ-approximations of all τ-significant frequencies, has polynomial costs in $\log N$, m, $1/\tau$ and $1/\delta$.

Finally, we mention sparse FFT algorithms based on Prony's method that are completely deterministic and can be operated with $\mathcal{O}(m^3)$ operations (independent of the signal size N), see e.g. [7, 13], but usually with an unstable numerical behavior. A better numerical performance can be achieved by randomization leading to a complexity $m^{5/3} \log^2 N$, see [7].

Compared to the approaches above, we restrict ourselves to signal vectors possessing a small support interval. Vectors with small support appear in different applications as e.g. in X-ray microscopy, where compact support is a frequently used a-priori condition in phase retrieval, as well as in computer tomography reconstructions.

Notations. First, we fix some notations. For a vector $\mathbf{x} \in \mathbb{C}^N$ of length N, let its discrete Fourier transform be defined by $\widehat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$, where the Fourier matrix $\mathbf{F}_N \in \mathbb{C}^{N \times N}$ is given by

$$\mathbf{F}_N := \left(\omega_N^{jk}\right)_{j,k=0}^{N-1}, \qquad \omega_N := \mathrm{e}^{\frac{-2\pi\,\mathrm{i}}{N}}.$$

Let the *support length* $m = |\operatorname{supp} \mathbf{x}|$ of $\mathbf{x} \in \mathbb{C}^N$ be defined as the minimal integer m for which there exists a $\mu \in \{0, \ldots, N-1\}$ such that the components x_k of $\mathbf{x}$ vanish for all $k \notin I := \{(\mu + r) \bmod N, \quad r = 0, \ldots, m-1\}$. The index set I is called *support interval* of $\mathbf{x}$. Obviously, the support length m of $\mathbf{x}$ is an upper bound for its sparsity $\|\mathbf{x}\|_0$, i.e., the number of nonzero components of $\mathbf{x}$, since there may be zero components of $\mathbf{x}$ inside the support interval I. However, we always have $x_\mu \neq 0$ and $x_{\mu+m-1} \neq 0$. Observe that the support length m of a vector $\mathbf{x} \in \mathbb{C}^N$ is always uniquely defined while the support interval itself resp. the first support index μ needs not to be unique. For example, considering the vector $\mathbf{x} = (x_k)_{k=0}^{N-1} \in C^N$ of even length N given by $x_1 = x_{N/2+1} = 1$ and $x_k = 0$ for $k \in \{0, \ldots, N-1\} \setminus \{1, N/2+1\}$, we find for $\mathbf{x}$ the support length $m = N/2+1$ while the support interval can be chosen either as $\{1, \ldots, N/2+1\}$ or as $\{N/2+1, \ldots, N-1, 0, 1\}$. However, if $m \leq \frac{N}{2}$, then the support interval and hence also the first support index μ are uniquely determined.

Our contribution. In this paper, we will derive new deterministic algorithms to reconstruct a vector $\mathbf{x} \in \mathbb{C}^N$ with support length $m < N$ from its discrete Fourier transform $\widehat{\mathbf{x}} \in \mathbb{C}^N$. In case of exact data, we will use less than $4m$ Fourier samples and $\mathcal{O}(m \log_2 m)$ arithmetical operations to recover $\mathbf{x}$. Thus, the algorithm already starts to pay off for $m < \frac{N}{4}$.

In case of noisy Fourier measurements, we will use $\mathcal{O}(m \log_2 N)$ arithmetical operations. More exactly, we will employ one or several FFTs of size less than $4m$ as well as the computation of $\lfloor \log_2 \frac{N}{m} \rfloor - 1$ scalar products of length m to recover the full vector in a stable way.

In both cases, for exact as well as for noisy measurements, the algorithms are based on the idea that for $N = 2^J$ the nonzero components of $\mathbf{x}$ can already be computed by evaluating a periodization of $\mathbf{x}$ with length $2^L \geq m$. Thus, for the complete reconstruction of $\mathbf{x}$, we only need to compute in a second step the correct support interval resp. the first support index of $\mathbf{x}$.

Organization of the paper. In Section 2, we recall some properties of the discrete Fourier transform that will be used in our approach. Section 3 is devoted to the sparse FFT algorithm for m-sparse vectors in case of exact Fourier measurements. Section 4 considers a more stable variant of the different steps of the algorithm that make the method robust in presence of noise, and even improves the accuracy of the resulting vector in comparison to the usual FFT method while using only $\mathcal{O}(m \log N)$ arithmetical operations. Finally, we present our numerical experiments in Section 5.

2 Preliminaries

Throughout the paper let $N := 2^J$ with some $J > 0$.

We consider the periodizations $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$ of $\mathbf{x}$,

$$\mathbf{x}^{(j)} = (x_k^{(j)})_{k=0}^{2^j-1} = \left(\sum_{\ell=0}^{2^{J-j}-1} x_{k+2^j\ell} \right)_{k=0}^{2^j-1} \tag{2.1}$$

for $j = 0, \ldots, J$. Obviously, $\mathbf{x}^{(0)} = \sum_{k=0}^{N-1} x_k$ is the sum of all components of $\mathbf{x}$, $\mathbf{x}^{(1)} = (\sum_{k=0}^{N/2-1} x_{2k}, \sum_{k=0}^{N/2-1} x_{2k+1})^T$ and $\mathbf{x}^{(J)} = \mathbf{x}$. We recall the following relationship for the discrete Fourier transform of the vectors $\mathbf{x}^{(j)}$, $j = 0, \ldots, J$, in terms of $\widehat{\mathbf{x}}$.

Lemma 2.1 *For the vectors $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$, $j = 0, \ldots, J$, in* (2.1)*, we have the discrete Fourier transform*

$$\widehat{\mathbf{x}}^{(j)} := \mathbf{F}_{2^j} \mathbf{x}^{(j)} = (\widehat{x}_{2^{J-j}k})_{k=0}^{2^j-1},$$

where $\widehat{\mathbf{x}} = (\widehat{x}_k)_{k=0}^{N-1} = \mathbf{F}_N \mathbf{x}$ is the Fourier transform of $\mathbf{x} \in \mathbb{C}^N$.

Proof: By definition, we find for the components $\widehat{x}_k^{(j)}$ of $\widehat{\mathbf{x}}^{(j)}$

$$\begin{aligned} \widehat{x}_k^{(j)} &:= \sum_{r=0}^{2^j-1} x_r^{(j)} \, \omega_{2^j}^{rk} = \sum_{r=0}^{2^j-1} \sum_{\ell=0}^{2^{J-j}-1} x_{r+2^j\ell} \, \omega_N^{2^{J-j}rk} \\ &= \sum_{n=0}^{N-1} x_n \, \omega_N^{2^{J-j}nk} = \widehat{x}_{2^{J-j}k}, \qquad k = 0, \ldots, 2^j - 1. \end{aligned}$$

Thus the assertion follows. ∎

Lemma 2.1 tells us that for a given vector $\widehat{\mathbf{x}}$ the Fourier transform of the periodized vectors $\widehat{\mathbf{x}}^{(j)}$ needs not to be computed but is just obtained by picking suitable components of $\widehat{\mathbf{x}}$. Further, we want to make use of the following simple observation.

Lemma 2.2 *Let $\mathbf{x} = (x_k)_{k=0}^{N-1} \in \mathbb{C}^N$, $N = 2^J$, and let $\mathbf{y} = (y_k)_{k=0}^{N-1} \in \mathbb{C}^N$ be its shifted version with components $y_k := x_{(k+2^j\nu) \mathrm{mod} N}$, $k = 0, \dots, N-1$, for some $j \in \{0, \dots, J-1\}$ and $\nu \in \{0, \dots, 2^{J-j} - 1\}$. Then the components of the Fourier transformed vectors $\widehat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$, $\widehat{\mathbf{y}} = \mathbf{F}_N \mathbf{y}$ satisfy*

$$\widehat{y}_l = \omega_{2^{J-j}}^{-l\nu}\, \widehat{x}_l, \qquad l = 0, \dots, N-1.$$

In particular, for $j = J-1$ and $\nu = 1$ we have $y_k = x_{(k+N/2) \mathrm{mod} N}$, $k = 0, \dots, N-1$, with

$$\widehat{x}_{2k} = \widehat{y}_{2k}, \qquad \widehat{x}_{2k+1} = -\widehat{y}_{2k+1}, \qquad k = 0, \dots, \frac{N}{2} - 1.$$

Proof: With $\widehat{\mathbf{x}} = \mathbf{F}_N \mathbf{x} = (\widehat{x}_l)_{l=0}^{N-1}$ and $\widehat{\mathbf{y}} = \mathbf{F}_N \mathbf{y} = (\widehat{y}_l)_{l=0}^{N-1}$ we obtain

$$\widehat{y}_l = \sum_{k=0}^{N-1} y_k\, \omega_N^{lk} = \sum_{k=0}^{N-1} x_{(k+2^j\nu) \mathrm{mod} N}\, \omega_N^{lk} = \sum_{k=0}^{N-1} x_k\, \omega_N^{l(k-2^j\nu)} = \omega_{2^{J-j}}^{-l\nu}\, \widehat{x}_l.$$

For $j = J-1$ and $\nu = 1$ the assertion follows with $\omega_{2^{J-j}}^{-l\nu} = \omega_2^{-l} = (-1)^l$. ∎

3 Reconstruction of x from exact Fourier data

We assume that the Fourier transformed vector $\widehat{\mathbf{x}} = \mathbf{F}_N \mathbf{x}$ is given, and that the support length m of $\mathbf{x}$ or an upper bound m of it is known a-priori. Now, we apply the following simple procedure to recover $\mathbf{x}$.

Let $2^{L-1} < m \le 2^L$. Then, by Lemma 1, $\widehat{\mathbf{x}}^{(L+1)} = (\widehat{x}_{2^{J-(L+1)}k})_{k=0}^{2^{L+1}-1}$ is the Fourier transform of $\mathbf{x}^{(L+1)}$. In a first step, we compute $\mathbf{x}^{(L+1)}$ using inverse FFT of length 2^{L+1}.

Since $|\mathrm{supp}\, \mathbf{x}| = m \le 2^L$ by assumption, it follows that $\mathbf{x}^{(L+1)}$ has already the same support length as $\mathbf{x}$ since for each $k \in \{0, \dots, 2^{L+1} - 1\}$ the sum in

$$x_k^{(L+1)} = \sum_{\ell=0}^{2^{J-L-1}-1} x_{k+2^{L+1}\ell} \tag{3.1}$$

contains at most one nonvanishing term. Moreover, also the support itself, or equivalently the first support index $\mu = \mu^{(L+1)}$ of $\mathbf{x}^{(L+1)}$, is uniquely determined.

Thus, in order to recover $\mathbf{x}$ from $\mathbf{x}^{(L+1)}$, we only need to compute the correct first support index $\mu^{(J)}$ of $\mathbf{x}$, such that the components of $\mathbf{x}$ are determined by

$$x_{(\mu^{(J)}+k) \mathrm{mod}\, N} = \begin{cases} x^{(L+1)}_{(\mu^{(L+1)}+k) \mathrm{mod}\, 2^{L+1}} & k = 0, \dots, m-1, \\ 0 & k = m, \dots, N-1. \end{cases}$$

From (3.1) we know that $\mu^{(J)}$ is of the form $\mu^{(J)} = \mu^{(L+1)} + 2^{L+1}\nu$ for some $\nu \in \{0, 1, \dots, 2^{J-L-1} - 1\}$.

Theorem 3.1 *Let* $\mathbf{x} \in \mathbb{C}^N$, $N = 2^J$, *have support length* m *(or a support length bounded by* m*) with* $2^{L-1} < m \leq 2^L$. *For* $L < J-1$, *let* $\mathbf{x}^{(L+1)}$ *be the* 2^{L+1}*-periodization of* $\mathbf{x}$. *Then* $\mathbf{x}$ *can be uniquely recovered from* $\mathbf{x}^{(L+1)}$ *and one nonzero component of the vector* $(\widehat{x}_{2k+1})_{k=0}^{N/2-1}$.

Proof: Let $\mathbf{x}^{(L+1)}$ have the support interval $\{(\mu^{(L+1)}+r) \bmod 2^{L+1},\ r = 0, \ldots, m-1\}$. Since $m \leq 2^L$, this support interval is uniquely determined. Further, we know that by construction the first index $\mu^{(J)}$ of the support interval of $\mathbf{x}$ has the form $\mu^{(J)} = \mu^{(L+1)} + 2^{L+1}\nu$ for some $\nu \in \{0, \ldots, 2^{J-L-1}-1\}$. We now consider the vector $\mathbf{u}^0 \in \mathbb{C}^N$ that is given by the components

$$u^0_{\mu^{(L+1)}+k} = \begin{cases} x^{(L+1)}_{(\mu^{(L+1)}+k) \bmod 2^{L+1}} & k = 0, \ldots, m-1, \\ 0 & k = m, \ldots, N-1. \end{cases}$$

The vector $\mathbf{u}^0$ is obtained as one possible vector in $\mathbb{C}^N$ with support length m possessing the 2^{L+1}-periodization $\mathbf{x}^{(L+1)}$. Obviously the first index of the support interval of $\mathbf{u}^0$ is $\mu^{(L+1)}$. Therefore, all further vectors in $\mathbb{C}^N$ with a support length m and possessing the 2^{L+1}-periodization $\mathbf{x}^{(L+1)}$ can be written as a shifted version of $\mathbf{u}^0$ of the form

$$\mathbf{u}^\nu := (u^\nu_k)_{k=0}^{N-1} \qquad \text{with} \qquad u^\nu_k = u^0_{(k+2^{L+1}\nu) \bmod N},\ k = 0, \ldots, N-1,$$

for some $\nu \in \{1, \ldots, 2^{J-L-1}-1\}$. Thus, the desired vector $\mathbf{x}$ is contained in the set $\{\mathbf{u}^\nu : \nu = 0, \ldots, 2^{J-L-1}-1\}$.

It remains to find ν such that $\mathbf{x} = \mathbf{u}^\nu$. From Lemma 2.2 it follows for the components of the Fourier transform $\widehat{\mathbf{u}}^\nu = (\widehat{u}^\nu_l)_{l=0}^{N-1}$ that $\widehat{u}^\nu_l = \omega_{2^{J-L-1}}^{-l\nu}\, \widehat{u}^0_l$.

Choosing now an odd-indexed nonzero Fourier value $\widehat{x}_{2k_0+1}$ of the given vector $\widehat{\mathbf{x}}$, we can compare it to the corresponding component of $\widehat{\mathbf{u}}^0$,

$$\widehat{u}^0_{2k_0+1} = \sum_{r=0}^{m-1} x^{(L+1)}_{(\mu^{(L+1)}+r) \bmod 2^{L+1}}\, \omega_N^{(\mu^{(L+1)}+r)(2k_0+1)}$$

and obtain the correct value ν from $\omega_{2^{J-L-1}}^{-(2k_0+1)\nu} = \widehat{x}_{2k_0+1}/\widehat{u}^0_{2k_0+1}$. We remark that $\nu \in \{0, \ldots, 2^{J-L-1}-1\}$ is indeed uniquely determined by $\omega_{2^{J-L-1}}^{-(2k_0+1)\nu}$ since $\gcd(2k_0+1, 2^{J-L-1}) = 1$, i.e., $2k_0+1$ and 2^{J-L-1} are mutually prime. Finally, the vector $\mathbf{x}$ is obtained as $\mathbf{x} = \mathbf{u}^\nu$. ■

We summarize the algorithm to evaluate $\mathbf{x}$ from $\widehat{\mathbf{x}}$ for exact Fourier data as follows.

Algorithm 3.2 *(Sparse FFT for vectors with small support for exact Fourier data)*
Input: $\widehat{\mathbf{x}} \in \mathbb{C}^N$, $N = 2^J$, $|\operatorname{supp} \mathbf{x}| \leq m < N$.

- *Compute* L *such that* $2^{L-1} < m \leq 2^L$, *i.e.,* $L := \lceil \log_2 m \rceil$.
- *If* $L = J$ *or* $L = J-1$, *compute* $\mathbf{x} = \mathbf{F}_N^{-1}\widehat{\mathbf{x}}$ *using an FFT algorithm of length* N.
- *If* $L < J-1$*:*

1. Choose $\widehat{\mathbf{a}} := (\widehat{x}_{2^{J-(L+1)}k})_{k=0}^{2^{L+1}-1}$ *and compute*

$$\mathbf{x}^{(L+1)} := \mathbf{F}_{2^{L+1}}^{-1}\widehat{\mathbf{a}}$$

using an FFT algorithm for the inverse discrete Fourier transform of length 2^{L+1}*.*

2. Determine the first support index $\mu^{(L+1)} \in \{0, \ldots, 2^{L+1}-1\}$ *of* $\mathbf{x}^{(L+1)}$ *such that* $x^{(L+1)}_{\mu^{(L+1)}} \neq 0$ *and* $x_k^{(L+1)} = 0$ *for* $k \notin \{(\mu^{(L+1)} + r) \bmod 2^{L+1}, r = 0, \ldots, m-1\}$*.*

3. Choose a Fourier component $\widehat{x}_{2k_0+1} \neq 0$ *of* $\widehat{\mathbf{x}}$ *and compute the sum*

$$\widehat{u}^0_{2k_0+1} := \sum_{\ell=0}^{m-1} x^{(L+1)}_{(\mu^{(L+1)}+\ell) \bmod 2^{L+1}} \, \omega_N^{(2k_0+1)(\mu^{(L+1)}+\ell)}.$$

4. Compute the quotient $b := \widehat{x}_{2k_0+1}/\widehat{u}^0_{2k_0+1}$ *that is by construction of the form* $b = \omega^p_{2^{J-L-1}}$ *for some* $p \in \{0, \ldots, 2^{J-L-1}-1\}$*, and find* $\nu \in \{0, \ldots, 2^{J-L-1}-1\}$ *such that* $(2k_0+1)\,\nu = p \bmod 2^{J-L-1}$*.*

5. Set $\mu^{(J)} := \mu^{(L+1)} + 2^{L+1}\nu$*, and* $\mathbf{x} := (x_k)_{k=0}^{N-1}$ *with entries*

$$x_{(\mu^{(J)}+\ell) \bmod N} := \begin{cases} x^{(L+1)}_{(\mu^{(L+1)}+\ell) \bmod 2^{L+1}} & \ell = 0, \ldots, m-1, \\ 0 & \ell = m, \ldots, N-1. \end{cases}$$

Output: $\mathbf{x}$.

We simply observe that the proposed algorithm has an arithmetical complexity of $\mathcal{O}(m \log m)$. In step 1 we employ an FFT algorithm of length $2^{L+1} < 4m$ having this complexity. All other steps can be performed with $\mathcal{O}(m)$ or less operations. Particularly, the support interval (resp. the first support index $\mu^{(L+1)}$ of $\mathbf{x}^{(L+1)}$) in step 2 can e.g. be found by computing the local energies $e_k := \sum_{\ell=k}^{m+k-1} |x^{(L+1)}_{\ell \bmod 2^{L+1}}|^2$ for $k = 0, \ldots, 2^{L+1}-1$, and taking $\mu^{(L+1)} := \operatorname{argmax}_k e_k$. Here, e_0 can be computed by at most $\mathcal{O}(m)$ operations, and all further energies by using the recursion $e_{k+1} = e_k - |x_k|^2 + |x_{k+m}|^2$, where we need to keep in mind that there are only m nonzero entries in $\mathbf{x}^{(L+1)}$.

The complete algorithm requires less then $4m$ Fourier values for the overall evaluation.

Let us give some further remarks on Algorithm 3.2.

Remark 3.3

It is always possible to choose a nonzero Fourier component of the vector $(\widehat{x}_{2k+1})_{k=0}^{N/2-1}$*. This can be seen as follows. Let* $\operatorname{supp}\mathbf{x} = \{\mu^{(J)}, \mu^{(J)}+1 \bmod N, \ldots, \mu^{(J)} + m - 1 \bmod N\}$ *be the support of* $\mathbf{x}$*, where* $m \leq 2^L \leq N/4$*. Then the trigonometric polynomial*

$$p(\omega) = \left| \sum_{k=0}^{N-1} x_k \, \mathrm{e}^{-\mathrm{i}\omega k} \right|^2 = \left| \mathrm{e}^{-\mathrm{i}\omega\mu^{(J)}} \sum_{\ell=0}^{m-1} x_{(\mu^{(J)}+\ell) \bmod N} \, \mathrm{e}^{-\mathrm{i}\omega\ell} \right|^2 \tag{3.2}$$

is real, non-negative and of degree $\leq m-1$*. Hence it possesses at most* $m-1$ *pairwise different zeros, where all zeros are at least double zeros. Observing now that* $|\widehat{x}_k|^2 = p\left(\frac{2\pi k}{N}\right)$*, we can conclude that not all* $\widehat{x}_{2k+1}$*,* $k = 0, \ldots, N/2-1$*, can be zeros of* $p(\omega)$ *since* $N/2 \geq 2m$*.*

Remark 3.4

As shown in the remark above, there can be at most $m-1$ zero components in the vector $(\widehat{x}_{2k+1})_{k=0}^{N/2-1}$. However, for a stable computation of the correct first support index $\mu^{(J)}$, the Fourier value $\widehat{x}_{2k_0+1}$ used in Algorithm 3.2 *should have large modulus. This may be ensured by taking*

$$k_0 := \operatorname{argmax}\{|\widehat{x}_{2k+1}|^2 : \, k = 0, \ldots, N/2-1\}.$$

Unfortunately, this procedure is quite costly. Instead, we may compute

$$k_0^{(L+1)} := \operatorname{argmax}\{|\widehat{x}_{2^{J-(L+1)}k}|^2 : \, k = 0, \ldots, 2^{L+1}-1\}.$$

Then we have for the trigonometric polynomial in (3.2),

$$|\widehat{x}_{2^{J-(L+1)}k_0^{(L+1)}}|^2 = p\left(\frac{2\pi k_0^{(L+1)}}{2^{L+1}}\right) = \max_{k=0,\ldots,2^{L+1}-1} p\left(\tfrac{2\pi k}{2^{L+1}}\right) > 0,$$

and it is likely that $p\left(\frac{2\pi k_0^{(L+1)}}{2^{L+1}}\right)$ is close to the global maximum $\|p\|_\infty$ of $p(\omega)$. In step 3 of Algorithm 3.2 *we may now choose the one neighboring Fourier value with maximal modulus, i.e., either $\widehat{x}_{2^{J-(L+1)}k_0^{(L+1)}+1}$ or $\widehat{x}_{2^{J-(L+1)}k_0^{(L+1)}-1}$.*

If $p(\omega)$ attains its global maximum at some point $\omega_0 \in \left[\frac{2\pi(k_0^{(L+1)}-1/2)}{2^{L+1}}, \frac{2\pi(k_0^{(L+1)}+1/2)}{2^{L+1}}\right)$, then it follows that $\left|\omega_0 - \frac{2\pi k_0^{(L+1)}}{2^{L+1}}\right| \le \frac{\pi}{2^{L+1}}$. With the above choice of $\widehat{x}_{2k_0+1}$ with $k_0 = 2^{J-L-2}k_0^{(L+1)}$ or $k_0 = 2^{J-L-2}k_0^{(L+1)}-1$, we can further assume that $\left|\omega_0 - \frac{2\pi(2k_0+1)}{N}\right| \le \frac{\pi}{2^{L+1}}$, and applying the Lemma of Stečkin (see e.g. [4]), we find

$$|\widehat{x}_{2k_0+1}|^2 = p\left(\tfrac{2\pi(2k_0+1)}{N}\right) \ge \|p\|_\infty \cos\left(\tfrac{\pi(m-1)}{2^{L+1}}\right) > 0.$$

4 Reconstruction of x from noisy Fourier data

Let us now assume that the given Fourier data are noisy, i.e., they are perturbed by uniform noise,

$$\widehat{y}_k = \widehat{x}_k + \epsilon_k \tag{4.1}$$

with $|\epsilon_k| \le \delta$. Similarly as before, we want to reconstruct $\mathbf{x}$ from the noisy Fourier vector $\widehat{\mathbf{y}}$ using the a-priori knowledge that $\mathbf{x}$ has a support interval of length $m < N$.

For that purpose, we propose a stabilized variant of Algorithm 3.2. The stabilization regards the following essential steps in the algorithm, namely
(1) the correct determination of the support interval of $\mathbf{x}^{(L+1)}$,
(2) the correct determination of the support interval resp. the first support index $\mu^{(J)}$ of the desired vector $\mathbf{x}$, and
(3) the evaluation of the nonzero components of $\mathbf{x}$ within the support interval that may be improved by employing more Fourier values $\widehat{\mathbf{y}}$ than in the case of exact data.

(1) Stable determination of the support interval of $\mathbf{x}^{(L+1)}$.

For that purpose we use the following observation.

Theorem 4.1 *Let $\mathbf{x} \in \mathbb{C}^N$, $N = 2^J$, have support length m (or a support length bounded by m) with $2^{L-1} < m \le 2^L$. For $L < J-1$, let $\mathbf{x}^{(L+1)} = (x_k^{(L+1)})_{k=0}^{2^{L+1}-1}$ be the 2^{L+1}-periodization of $\mathbf{x}$ and $\widehat{\mathbf{x}}^{(L+1)} = (\widehat{x}_{2^{J-L-1}k})_{k=0}^{2^{L+1}-1}$ its Fourier transform. Then, for each shifted vector of the form*

$$\widehat{\mathbf{z}}^{(\kappa)} := \big(\widehat{x}_{2^{J-L-1}k+\kappa}\big)_{k=0}^{2^{L+1}-1}, \qquad \kappa = 0, \ldots, 2^{J-L-1}-1,$$

the inverse Fourier transform $\mathbf{z}^{(\kappa)} = (z_\ell^{(\kappa)})_{\ell=0}^{2^{L+1}-1} = \mathbf{F}_{2^{L+1}}^{-1}\widehat{\mathbf{z}}^{(\kappa)}$ satisfies

$$|z_\ell^{(\kappa)}| = |x_\ell^{(L+1)}| \quad \ell = 0, \ldots, 2^{L+1}-1.$$

Proof: By definition, we obtain for the components $z_\ell^{(\kappa)}$ of $\mathbf{z}^{(\kappa)}$

$$\begin{aligned} z_\ell^{(\kappa)} &= \frac{1}{2^{L+1}} \sum_{k=0}^{2^{L+1}-1} \widehat{x}_{2^{J-L-1}k+\kappa}\, \omega_{2^{L+1}}^{-k\ell} \\ &= \frac{1}{2^{L+1}} \sum_{k=0}^{2^{L+1}-1} \sum_{r=0}^{N-1} x_r\, \omega_N^{r(2^{J-L-1}k+\kappa)}\, \omega_{2^{L+1}}^{-k\ell} \\ &= \frac{1}{2^{L+1}} \sum_{r=0}^{N-1} x_r\, \omega_N^{r\kappa} \sum_{k=0}^{2^{L+1}-1} \omega_{2^{L+1}}^{k(r-\ell)} \\ &= \sum_{j=0}^{2^{J-L-1}-1} x_{\ell+2^{L+1}j}\, \omega_N^{(\ell+2^{L+1}j)\kappa} = \omega_N^{\ell\kappa} \sum_{j=0}^{2^{J-L-1}-1} x_{\ell+2^{L+1}j}\, \omega_N^{2^{L+1}j\kappa}. \end{aligned}$$

Since $|\operatorname{supp}\mathbf{x}| = m < 2^{L+1}$, for each ℓ the above sum contains only one value, thus $|z_\ell^{(\kappa)}| = |x_\ell^{(L+1)}|$. ∎

Obviously, we have $\mathbf{z}^{(0)} = \mathbf{x}^{(L+1)}$ by definition. We observe that all vectors $\mathbf{z}^{(\kappa)}$ are constructed from different Fourier components of $\widehat{\mathbf{x}}$. This circumstance will be used to stabilize the algorithm. Applying the noisy measurements $\widehat{y}_k$ in (4.1) instead of $\widehat{x}_k$, let $\widehat{\widetilde{\mathbf{z}}}^{(\kappa)} := (\widehat{y}_{2^{J-L-1}k+\kappa})_{k=0}^{2^{L+1}-1}$. As before in Algorithm 3.2, we compute in a first step the vector $\widetilde{\mathbf{z}}^{(0)} = \mathbf{y}^{(L+1)}$ from the noisy measurements $(\widehat{y}_{2^{J-(L+1)}k})_{k=0}^{2^{L+1}-1}$. In order to determine the support interval of $\mathbf{x}^{(L+1)}$ from the vector $\widetilde{\mathbf{z}}^{(0)}$ we consider the energies

$$\widetilde{e}_k^{(0)} := \sum_{\ell=k}^{m+k-1} |y_{\ell\,\mathrm{mod} 2^{L+1}}^{(L+1)}|^2 = \sum_{\ell=k}^{m+k-1} |\widetilde{z}_{\ell \bmod 2^{L+1}}^{(0)}|^2, \qquad k = 0, \ldots, 2^{L+1}-1,$$

and take $\mu_0^{(L+1)} := \underset{k}{\operatorname{argmax}}\, \widetilde{e}_k^{(0)}$ as the first estimate for $\mu^{(L+1)}$. For higher noise levels, we stabilize the computation of $\mu^{(L+1)}$ as follows: We compute also the vector $\widetilde{\mathbf{z}}^{(2^{J-L-2})}$ by applying the inverse FFT to $\widehat{\widetilde{\mathbf{z}}}^{(2^{J-L-2})} = (\widehat{y}_{2^{J-L-2}(2k+1)})_{k=0}^{2^{L+1}-1}$, compute the energies

$$\widetilde{e}_k^{(1)} := \sum_{\ell=k}^{m+k-1} |\widetilde{z}_{\ell \bmod 2^{L+1}}^{(2^{J-L-2})}|^2, \qquad k = 0, \ldots, 2^{L+1}-1,$$

and take $\mu_1^{(L+1)} := \underset{k}{\operatorname{argmax}} \frac{1}{2}(\widetilde{e}_k^{(0)} + \widetilde{e}_k^{(1)})$. If $\mu_1^{(L+1)} = \mu_0^{(L+1)}$, then this index is taken as the first support index of $\mathbf{x}^{(L+1)}$. Otherwise, we compute a further vector, e.g., $\widetilde{\mathbf{z}}^{(2^{J-L-3})} = \mathbf{F}_{2^{L+1}}^{-1}(\widehat{y}_{2^{J-L-3}(4k+1)})_{k=0}^{2^{L+1}-1}$, evaluate the corresponding energies $\widetilde{e}_k^{(2)}$ and $\mu_2^{(L+1)} := \underset{k}{\operatorname{argmax}} \frac{1}{3}(\widetilde{e}_k^{(0)} + \widetilde{e}_k^{(1)} + \widetilde{e}_k^{(2)})$, etc.

(2) Stable determination of the support interval of x.

In order to stabilize the computation of the first support index $\mu^{(J)}$ of the complete vector $\mathbf{x} = \mathbf{x}^{(J)}$ resp. the shift ν such that $\mu^{(J)} = \mu^{(L+1)} - 2^{L+1}\nu$, we employ an iterative procedure, where at each iteration step, the support interval of the periodized vector $\mathbf{x}^{(j+1)}$, $j = L+1, \dots J-1$, is computed from the support interval of the vector $\mathbf{x}^{(j)}$ of half length.

At each iteration step we apply the following procedure. Observing that $\mathbf{x}^{(j+1)}$ has the same support length m as $\mathbf{x}^{(j)}$, and using the relation

$$x_k^{(j+1)} + x_{k+2^j}^{(j+1)} = x_k^{(j)}, \qquad k = 0, \dots, 2^j - 1,$$

it is sufficient to check whether $\mu^{(j+1)} = \mu^{(j)}$ or $\mu^{(j+1)} = \mu^{(j)} + 2^j$, i.e., whether the first support index $\mu^{(j)}$ of $\mathbf{x}^{(j)}$ is equal to the first support index $\mu^{(j+1)}$ of $\mathbf{x}^{(j+1)}$, or if the support has to be shifted by 2^j. We illustrate the two cases in detail in Figure 1.

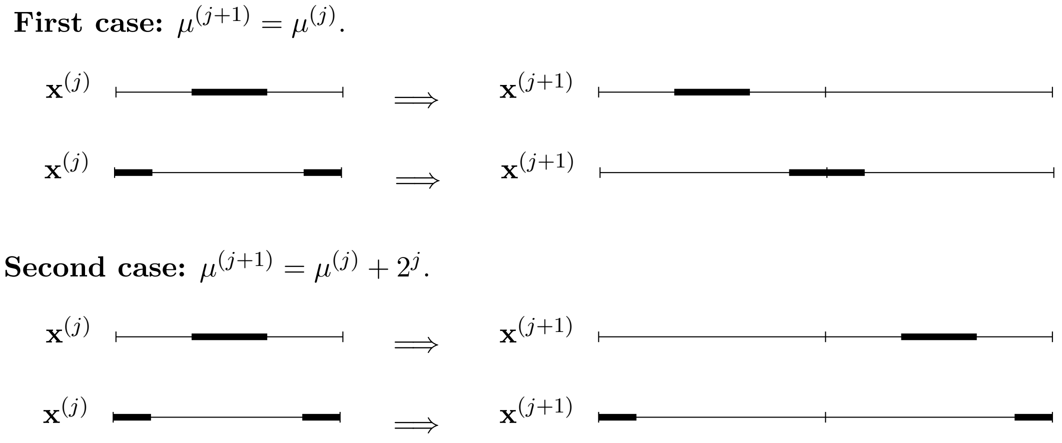


Figure 1:
Possible support change in one iteration step.

Generally, in order to recover $\mathbf{x}^{(j+1)}$ from $\mathbf{x}^{(j)}$, we apply the following procedure.

Theorem 4.2 *Let $\mathbf{x} \in \mathbb{C}^N$, $N = 2^J$, have support of length m (or a support length bounded by m) with $2^{L-1} < m \leq 2^L$. Further, let $\mathbf{x}^{(j)}$, $L+1 \leq j \leq J-1$, be the periodizations of $\mathbf{x} = \mathbf{x}^{(J)}$ as given in* (2.1). *Then, for each $j = L+1, \dots, J-1$, the vector $\mathbf{x}^{(j+1)}$ can be uniquely recovered from $\mathbf{x}^{(j)}$ and one nonzero component of the vector of Fourier values $\mathbf{y} := (\widehat{x}_{2^{J-(j+1)}(2k+1)})_{k=0}^{2^j-1}$.*

Proof: Let $\mathbf{x}^{(j)}$ be the given vector of length 2^j with support $\operatorname{supp}\mathbf{x}^{(j)} = \{(\mu^{(j)} + r) \bmod 2^j, \quad r = 0, \dots, n-1\}$. In order to determine $\mathbf{x}^{(j+1)}$, we only need to decide whether $\mu^{(j+1)} = \mu^{(j)}$ or $\mu^{(j+1)} = \mu^{(j)} + 2^j$, or equivalently, whether $\mathbf{x}^{(j+1)}$ is given by

$$x^{(j+1)}_{\mu^{(j)}+\ell} = \begin{cases} x_{(\mu^{(j)}+\ell) \bmod 2^j} & \ell = 0, \ldots, n-1, \\ 0 & \text{else}, \end{cases} \tag{4.2}$$

or by

$$x^{(j+1)}_{(\mu^{(j)}+2^j+\ell) \bmod 2^{j+1}} = \begin{cases} x_{(\mu^{(j)}+\ell) \bmod 2^j} & \ell = 0, \ldots, n-1, \\ 0 & \text{else}. \end{cases} \tag{4.3}$$

The two possible solutions $\mathbf{x}^{(j+1)}$ only differ by a shift of all components by 2^j. For simplicity let us denote the two solutions by $\mathbf{u}^{(0)}$ and $\mathbf{u}^{(1)}$. Then, according to Lemma 2.2, the Fourier transformed vectors $\widehat{\mathbf{u}}^{(0)} = (\widehat{u}^{(0)}_k)_{k=0}^{2^{j+1}-1}$ and $\widehat{\mathbf{u}}^{(1)} = (\widehat{u}^{(1)}_k)_{k=0}^{2^{j+1}-1}$ satisfy the relationship

$$\widehat{u}^{(0)}_{2k+1} = -\widehat{u}^{(1)}_{2k+1}, \qquad k = 0, \ldots, 2^j - 1.$$

Using now one nonzero Fourier value $\widehat{x}_{2^{J-(j+1)}(2k_0+1)} = \widehat{x}^{(j+1)}_{2k_0+1}$, we can determine the correct vector $\mathbf{x}^{(j+1)}$. For that purpose, we just compute

$$\widehat{u}^{(0)}_{2k_0+1} = \sum_{\ell=0}^{m-1} x^{(j)}_{(\mu^{(j)}+\ell) \bmod 2^j} \omega_{2^{j+1}}^{(2k_0+1)(\ell+\mu^{(j)})}$$

and compare it to the Fourier value $\widehat{x}_{2^{J-(j+1)}(2k_0+1)}$. If $|\widehat{u}^{(0)}_{2k_0+1} - \widehat{x}_{2^{J-(j+1)}(2k_0+1)}| < |\widehat{u}^{(0)}_{2k_0+1} + \widehat{x}_{2^{J-(j+1)}(2k_0+1)}|$, then $\mathbf{x}^{(j+1)} = \mathbf{u}^{(0)}$ and $\mu^{(j+1)} = \mu^{(j)}$, otherwise, we find $\mathbf{x}^{(j+1)} = \mathbf{u}^{(1)}$ and $\mu^{(j+1)} = \mu^{(j)} + 2^j$. ■

(3) Evaluation of the nonzero components of x.

Finally, we can use the vectors $\widetilde{\mathbf{z}}^{(\kappa)} \in \mathbb{C}^{2^{L+1}}$ computed in step (1) also for a more exact evaluation of the nonzero components x_k of $\mathbf{x}$ as follows. First, we employ our investigations in Theorem 4.1 and observe that

$$\widetilde{z}^{(\kappa)}_{\ell \bmod 2^{L+1}} = \omega_N^{\ell\kappa} \, (x_{(\ell+2^{L+1}\nu) \bmod N} \, \omega_N^{2^{L+1}\nu\kappa}), \qquad \ell = \mu^{(L+1)}, \ldots, \mu^{(L+1)} + m - 1.$$

Using the equation $\mu^{(J)} = \mu^{(L+1)} + 2^{L+1}\nu$, we can reformulate this as

$$\widetilde{z}^{(\kappa)}_{(\mu^{(L+1)}+k) \bmod 2^{L+1}} = x_{(\mu^{(J)}+k) \bmod N} \, \omega_N^{\kappa(\mu^{(J)}+k)}, \qquad k = 0, \ldots, m-1.$$

Therefore, if we have to compute the support entries $x_{(\mu^{(J)}+k) \bmod N}$, $k = 0, \ldots, m-1$, from the noisy measurements $(\widehat{y}_k)_{k=0}^{N-1}$ of $\widehat{\mathbf{x}}$, we can take the average

$$x_{(\mu^{(J)}+k) \bmod N} = \frac{1}{B+1} \sum_{r=0}^{B} \widetilde{z}^{(\kappa_r)}_{(\mu^{(L+1)}+k) \bmod 2^{L+1}} \, \omega_N^{-\kappa_r(\mu^{(J)}+k)}, \qquad k = 0, \ldots, m-1,$$

where $B+1$ is the number of the vectors $\widetilde{\mathbf{z}}^{(\kappa)} = \mathbf{F}^{-1}_{2^{L+1}} (\widehat{y}_{2^{J-L-1}k+\kappa})_{k=0}^{2^{L+1}-1}$ that we want to involve.

The complete algorithm to compute $\mathbf{x}$ from its Fourier transform by a fast sparse FFT algorithm can now be summarized as follows.

Algorithm 4.3 *(Sparse FFT for vectors with small support for noisy Fourier data)*
Input: *noisy measurement vector* $\widehat{\mathbf{y}} \in \mathbb{C}^N$, $N = 2^J$, $|\mathrm{supp}\, \mathbf{x}| \le m < N$.

- *Compute* L *such that* $2^{L-1} < m \le 2^L$, *i.e.,* $L := \lceil \log_2 m \rceil$.
- *If* $L = J$ *or* $L = J-1$, *compute* $\mathbf{x} = \mathbf{F}_N^{-1} \widehat{\mathbf{y}}$ *by inverse FFT.*
- *If* $L < J-1$*:*
  1. *Choose* $\widehat{\widetilde{\mathbf{z}}}^{(0)} := \widehat{\mathbf{y}}^{(L+1)} = (\widehat{y}_{2^{J-(L+1)}k})_{k=0}^{2^{L+1}-1}$ *and compute* $\widetilde{\mathbf{z}}^{(0)} = \mathbf{y}^{(L+1)} := \mathbf{F}_{2^{L+1}}^{-1} \widehat{\mathbf{y}}^{(L+1)}$ *using an FFT algorithm for the inverse discrete Fourier transform.*
  2. *Determine the first support index* $\mu^{(L+1)} \in \{0, \ldots, 2^{L+1}-1\}$ *of* $\mathbf{x}^{(L+1)}$ *using the following iteration:*
     - *Compute the energies*
     $$\widetilde{e}_k^{(0)} := \sum_{\ell=k}^{m+k-1} |\widetilde{z}^{(0)}_{\ell \bmod 2^{L+1}}|^2, \qquad k = 0, \ldots, 2^{L+1}-1,$$
     and compute $\mu_0^{(L+1)} := \operatorname*{argmax}_k \widetilde{e}_k^{(0)}$.
     - *Compute* $\widetilde{\mathbf{z}}^{(J-L-2)}$ *by IFFT from* $(\widehat{y}_{2^{J-L-2}(2k+1)})_{k=0}^{2^{L+1}-1}$, *determine*
     $$\widetilde{e}_k^{(1)} := \sum_{\ell=k}^{m+k-1} |\widetilde{z}^{(2^{J-L-2})}_{\ell \bmod 2^{L+1}}|^2, \qquad k = 0, \ldots, 2^{L+1}-1,$$
     and take $\mu_1^{(L+1)} := \operatorname*{argmax}_k \frac{1}{2}(\widetilde{e}_k^{(0)} + \widetilde{e}_k^{(1)})$.
     - *Set* $j := 0$.
     While $\mu_j^{(L+1)} \neq \mu_{j+1}^{(L+1)}$
     proceed by computing for a further $\kappa \in \{1, \ldots, 2^{J-L-1}-1\} \setminus \{2^{J-L-2}\}$ *the vector* $\widetilde{\mathbf{z}}^{(\kappa)}$ *by IFFT from* $(\widehat{y}_{2^{J-L-1}k+\kappa})_{k=0}^{2^{L+1}-1}$, *the energies*
     $$\widetilde{e}_k^{(j+2)} := \sum_{\ell=k}^{m+k-1} |\widetilde{z}^{(\kappa)}_{\ell \bmod 2^{L+1}}|^2, \qquad k = 0, \ldots, 2^{L+1}-1,$$
     and take $\mu_{j+2}^{(L+1)} := \operatorname*{argmax}_k \frac{1}{j+3} \sum_{r=0}^{j+2} \widetilde{e}_k^{(r)}$.
     Set $\mu^{(L+1)} := \mu_{j+2}^{(L+1)}$ *and* $j := j+1$.
     End (while).
     Set $\mathbf{x}^{(L+1)} = (x_k^{(L+1)})_{k=0}^{2^{L+1}-1}$ *with*
     $$x^{(L+1)}_{(k+\mu^{(L+1)}) \bmod 2^{L+1}} := \begin{cases} \widetilde{z}^{(0)}_{(k+\mu^{(L+1)}) \bmod 2^{L+1}} & k = 0, \ldots, m-1, \\ 0 & k = m, \ldots, 2^{L+1}-1. \end{cases}$$
  3. *For* $j = L+1, \ldots, J-1$
  Choose a Fourier component $\widehat{y}_{2^{J-(j+1)}(2k_0+1)} = \widehat{y}^{(j+1)}_{2k_0+1} \neq 0$ *and compute*
  $$a_{j+1} := \sum_{\ell=0}^{m-1} x^{(L+1)}_{(\mu^{(L+1)}+\ell) \bmod 2^{L+1}} \, \omega_{2^{j+1}}^{(2k_0+1)(\mu^{(j)}+\ell)}.$$

If $|a_{j+1} - \widehat{y}^{(j+1)}_{2k_0+1}| < |a_{j+1} + \widehat{y}^{(j+1)}_{2k_0+1}|$, *then set* $\mu^{(j+1)} := \mu^{(j)}$ *and* $\mathbf{x}^{(j+1)} := (x^{(j+1)}_k)^{2^{j+1}-1}_{k=0}$ *with entries*

$$x^{(j+1)}_{\mu^{(j)}+\ell} = \begin{cases} x^{(j)}_{(\mu^{(j)}+\ell) \bmod 2^j} & \ell = 0, \dots, n-1, \\ 0 & \text{else.} \end{cases}$$

If $|a_{j+1} - \widehat{y}^{(j+1)}_{2k_0+1}| \geq |a_{j+1} + \widehat{y}^{(j+1)}_{2k_0+1}|$, *then set* $\mu^{(j+1)} := \mu^{(j)} + 2^j$ *and* $\mathbf{x}^{(j+1)} := (x^{(j+1)}_k)^{2^{j+1}-1}_{k=0}$ *with entries*

$$x^{(j+1)}_{(\mu^{(j)}+2^j+\ell) \bmod 2^{j+1}} = \begin{cases} x^{(j)}_{(\mu^{(j)}+\ell) \bmod 2^j} & \ell = 0, \dots, n-1, \\ 0 & \text{else.} \end{cases}$$

4. *Assuming that* $\widetilde{\mathbf{z}}^{(\kappa_r)} \in \mathbb{C}^{2^{L+1}}$, $r = 0, \dots, B$, *have been evaluated already in step 2, we compute*

$$x_{(\ell+2^{L+1}\nu) \bmod N} = \frac{1}{B+1} \sum_{r=0}^{B} \widetilde{z}^{(\kappa_r)}_\ell \, \omega_N^{-\kappa_r(\ell+2^{L+1}\nu)},$$

$\ell = \mu^{(L+1)}, \dots, \mu^{(L+1)} + m - 1.$

Output: $\mathbf{x}^{(J)} = \mathbf{x}$.

Let us shortly summarize the arithmetical complexity of the algorithm. Step 1 and 2 together require two or more inverse FFTs of length $2^{L+1} < 4n$, i.e. $\mathcal{O}(m \log m)$ arithmetical operations. The number of involved vectors $\widetilde{\mathbf{z}}^{(\kappa)}$ depends on the noise level. The evaluation of energies can be done at each iteration step by $\mathcal{O}(m)$ operations.

In step 3, $J - (L+1) = \log_2 N - \lceil \log_2 m \rceil - 1 < \log_2(N/m)$ iterations are needed, where at each iteration a scalar product of length m has to be computed and to be compared to a given Fourier value. To find a nonzero Fourier value needs less than m comparisons, see also Remark 3.4. Hence only $\mathcal{O}(m \log m + m \log(N/m)) = \mathcal{O}(m \log_2 N)$ arithmetical operations are needed to recover $\mathbf{x}$ in the case of noisy data.

Remark 4.4

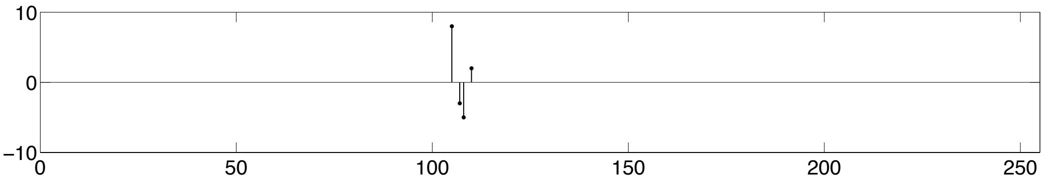
For the computation of $\mathbf{x}$ *it is sufficient to compute* $\mathbf{x}^{(L+1)}$ *as well as the first support index* $\mu^{(J)}$, *where* $\mu^{(j+1)}$, $j = L+1, \dots, J-1$, *is iteratively computed from* $\mu^{(j)}$. *The values* a_{j+1} *can be obtained directly from* $\mathbf{x}^{(L+1)}$ *and* $\mu^{(j)}$. *Hence, there is no reason to compute the intermediate vectors* $\mathbf{x}^{(j+1)}$ *in step 3 of Algorithm 4.3, they are only given for better illustration.*

5 Numerical results

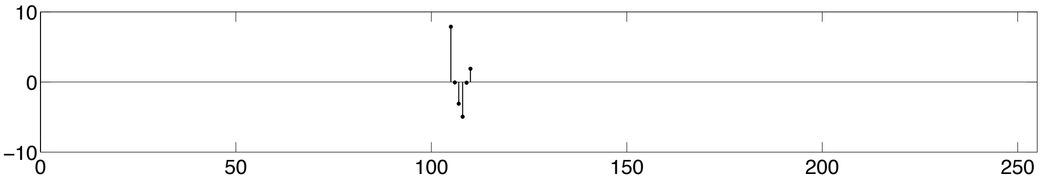
In this section, we discuss the behavior of the algorithm for noisy input data. We reconstruct randomly generated vectors $\mathbf{x}$ from disturbed Fourier data $\widehat{\mathbf{y}} = \widehat{\mathbf{x}} + \boldsymbol{\varepsilon}$, where we assume uniform noise $\boldsymbol{\varepsilon} = (\varepsilon_k)^{N-1}_{k=0}$ with $|\varepsilon_k| \leq \delta$ at different noise levels. As a noise measure, we use the SNR value

$$\text{SNR} = 20 \cdot \log_{10} \frac{\|\widehat{\mathbf{x}}\|_2}{\|\boldsymbol{\varepsilon}\|_2}.$$

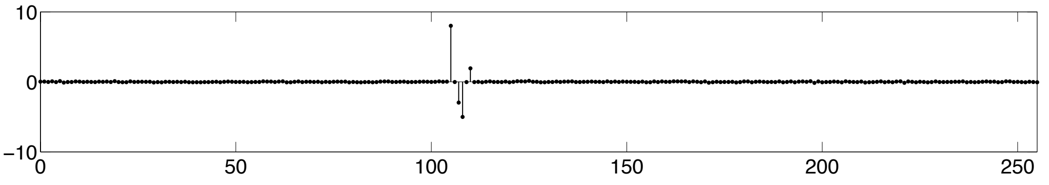
Let us first illustrate the algorithm for a vector $\mathbf{x}$ of length $N = 2^8 = 256$ and support length $m = 6$, i.e., we have $J = 8$ and $L = 3$. The nonzero entries of $\mathbf{x}$ are $x_{105} = 8$, $x_{107} = -3$, $x_{108} = -5$ and $x_{110} = 2$. We add a noise vector $\boldsymbol{\varepsilon}$ of SNR $= 20$ to $\widehat{\mathbf{x}}$ and reconstruct $\mathbf{x}$ from $\widehat{\mathbf{y}} = \widehat{\mathbf{x}} + \boldsymbol{\varepsilon}$. For the noise vector $\boldsymbol{\varepsilon}$ in this example, it holds that $\|\boldsymbol{\varepsilon}\|_\infty = 1.721$ and $\|\boldsymbol{\varepsilon}\|_1/256 = 0.952$. In Figure 2 we present $\mathbf{x}$, and we compare the reconstruction results of our algorithm to the result of the inverse Fourier transform applied to $\widehat{\mathbf{y}}$. The reconstruction $\mathbf{x}'$ computed by our deterministic sparse FFT algorithm has nonzero components $x'_{105} = 7.884+0.131\,\mathrm{i}$, $x'_{106} = -0.070-0.149\,\mathrm{i}$, $x'_{107} = -3.100 + 0.171\,\mathrm{i}$, $x'_{108} = -4.955 + 0.094\,\mathrm{i}$, $x'_{109} = -0.105 + 0.022\,\mathrm{i}$, $x'_{110} = 1.879 - 0.076\,\mathrm{i}$, and an error $\|\mathbf{x} - \mathbf{x}'\|_2/256 = 0.00146$ whereas the error by the inverse FFT is $\|\mathbf{x} - \mathbf{F}_{256}^{-1}\widehat{\mathbf{y}}\|_2/256 = 0.00395$. In this example, the reconstruction by our algorithm requires no additional vectors for the determination of $\mu^{(L+1)}$ in step 2 of Algorithm 4.3, i.e., we only use the two vectors $\widetilde{\mathbf{z}}^{(0)}$ and $\widetilde{\mathbf{z}}^{(J-L-2)} = \widetilde{\mathbf{z}}^{(3)}$ of length $2^{L+1} = 16$ in this step. Thus, we have taken $32 + 4 = 36$ Fourier values to recover $\mathbf{x}$.



(a)



(b)



(c)

Figure 2:
(a) Original vector $\mathbf{x}$ of length $N = 256$; (b) Reconstruction of $\mathbf{x}$ using the sparse FFT Algorithm 4.3; (c) Reconstruction of $\mathbf{x}$ using the inverse FFT.

In Figure 3 we show the reconstruction error using the sparse FFT Algorithm 4.3 for vectors $\mathbf{x}$ of length $N = 2^{22}$ and support length $m = 50$ where the vectors are randomly generated with $|\mathrm{Re}(x_k)| \le 10$, $|\mathrm{Im}(x_k)| \le 10$ for k in the support interval. We consider noisy Fourier data with SNR values between 0 and 50 and compute the reconstruction $\mathbf{x}'$ for 100 vectors $\mathbf{x}$ at different noise levels. The quality of the reconstructed vectors $\mathbf{x}'$ is evaluated by computing the norm $\|\mathbf{x} - \mathbf{x}'\|_2/N$. Figure 3 shows the average error norm over all 100 considered vectors. We compare the reconstruction results of our algorithm to the results of an inverse Fourier transform applied to the noisy vectors $\widehat{\mathbf{y}}$. For noise levels SNR ≤ 10, our algorithm made use of additional vectors $\widetilde{\mathbf{z}}^{(\kappa)}$ in step 2 of Algorithm 4.3 in order to improve the identification of the first support index $\mu^{(L+1)}$ of $\mathbf{x}^{(L+1)}$. Here, the algorithm evaluated at most five additional vectors (for SNR = 0), hence a total number of up to seven vectors $\widetilde{\mathbf{z}}^{(\kappa)}$ has been used to determine $\mu^{(L+1)}$ in some cases.

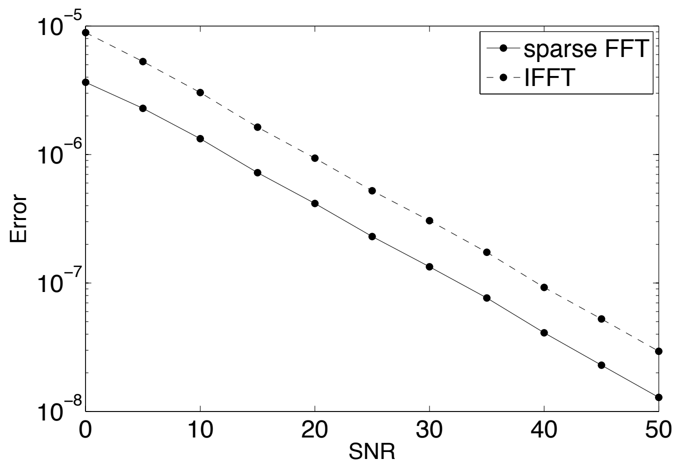


Figure 3:
Average reconstruction error $\|\mathbf{x} - \mathbf{x}'\|_2/N$ for different levels of uniform noise, comparing our deterministic sparse FFT algorithm and inverse FFT.

Determining the first index of the support of $\mathbf{x}$ (i.e., finding the correct $\mu = \mu^{(J)}$) is one of the crucial points of the algorithm for noisy input data. If $\mu^{(L+1)}$ is not identified correctly, the correct support interval cannot be found anymore, even if all shifts in step 3 of Algorithm 4.3 are correct.

In a third experiment we again take randomly generated vectors $\mathbf{x}$ of length $N = 2^{22}$ with support length $m = 50$ or $m = 2^{18}$, where $|\mathrm{Re}(x_k)| \le 10$, $|\mathrm{Im}(x_k)| \le 10$ for k in the support interval.

SNR	$N = 2^{22}$, $m = 50$			$N = 2^{22}$, $m = 2^{18}$		
	correctly identified μ	$\|\varepsilon\|_\infty$	$\|\varepsilon\|_1/N$	correctly identified μ	$\|\varepsilon\|_\infty$	$\|\varepsilon\|_1/N$
0	86%	68.367	37.002	78%	4927.294	2666.891
5	97%	37.799	20.458	93%	2911.275	1575.695
10	99%	22.262	12.049	97%	1365.686	739.186
15	100%	12.559	6.798	100%	841.737	455.585
20	100%	6.751	3.654	100%	483.223	261.542
25	100%	3.796	2.055	100%	261.897	141.752
30	100%	2.147	1.162	100%	144.593	78.259
35	100%	1.242	0.672	100%	84.374	45.665
40	100%	0.668	0.362	100%	47.666	25.800

Table 1:
Percentage of correctly identified μ and average norm of noise in 100 randomly chosen vectors for different noise levels, dependent on length N and support length m of $\mathbf{x}$.

Table 1 shows the number of cases in which the first support index μ has been determined correctly by Algorithm 4.3 for 100 randomly chosen vectors for each noise level. Additionally, we give an average for the norm of the noise vectors $\boldsymbol{\varepsilon}$ at each noise level.

The results show that the algorithm succeeds for very short support intervals as well as for long support intervals compared to the full vector length. In cases where μ could not be determined correctly, the error originates from step 2 of the algorithm where $\mu^{(L+1)}$ has to be determined. However, the deviation from the correct support was small (≤ 6) in any case.

We can conclude that even for high noise level, the support of the reconstructed vector $\mathbf{x}$ is correctly found in most of the cases. The support is always correct for noise levels with SNR ≥ 15. Table 1 also shows that the absolute noise ε_k at each component can be considerably large. It is essentially larger for vectors with larger support since in this case also the signal energy grows accordingly.

Acknowledgement

The research in this paper is partially funded by the project PL 170/16-1 of the German Research Foundation (DFG). This is gratefully acknowledged.

References

[1] A. Akavia, Deterministic sparse Fourier approximation via fooling arithmetic progressions, in Proc. 23rd COLT, 2010, pp. 381–393.

[2] A. Akavia, Deterministic sparse Fourier approximation via approximating arithmetic progressions, IEEE Trans. Inform. Theory **60**(3) (2014), 1733–1741.

[3] A. Gilbert, P. Indyk, M.A. Iwen, and L. Schmidt, Recent developments in the sparse Fourier transform, IEEE Signal Processing Magazine **31**(5) (2014), 91–100.

[4] J.J. Green, Calculating the maximum modulus of a polynomial using Stečkin's Lemma, SIAM J. Numer. Anal. **36**(4) (1999), 1022-1029.

[5] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, Near-optimal algorithm for sparse Fourier transform, Proc. 44th annual ACM symposium on Theory of Computing, 2012, pp. 563–578.

[6] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, Simple and practical algorithm for sparse Fourier transform, Proc. 23th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '12), 2012, pp. 1183–1194.

[7] S. Heider, S. Kunis, D. Potts, and M. Veit, A sparse Prony FFT, Proc. 10th International Conference on Sampling Theory and Applications (SAMPTA), 2013, pp. 572–575.

[8] P. Indyk, M. Kapralov, and E. Price, (Nearly) sample-optimal Fourier transform, Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 14), 2014, pp. 480–499.

[9] M.A. Iwen, Combinatorial sublinear-time Fourier algorithms, Found. Comput. Math. **10** (2010), 303–338.

[10] M.A. Iwen, Improved approximation guarantees for sublinear-time Fourier algorithms, Appl. Comput. Harmon. Anal. **34** (2013), 57–82.

[11] D. Lawlor, Y. Wang, and A. Christlieb, Adaptive sub-linear time Fourier algorithms, Advances in Adaptive Data Analysis **5**(1) (2013), 1350003 (25 pages).

[12] S. Pawar and K. Ramchandran, Computing a k-sparse n-length discrete Fourier transform using at most $4k$ samples and $\mathcal{O}(k \log k)$ complexity, IEEE International Symposium on Information Theory (2013), pp. 464–468.

[13] G. Plonka and M. Tasche, Prony methods for recovery of structured functions, GAMM-Mitt. **37**(2) (2014), 239–258.

[14] J. Schumacher, High performance sparse fast Fourier transform, Master Thesis, Computer Science. ETH Zürich, Switzerland, 2013.